TagParser: combiner un corpus annoté avec un corpus brut

Gil Francopoulo
Tagmatica
126 rue de Picpus 75012 Paris
www.tagmatica.com

Résumé

L'article se situe dans le contexte d'une recherche appliquée du secteur privé. Après avoir constaté que la combinaison d'un bon lexique avec un bon analyseur ne suffit pas à constituer un analyseur industriel opérationnel, nous avons cherché des solutions efficaces pour aller un peu plus loin. Nous montrons qu'il faut d'une part confronter la chaîne de traitement à des corpus réels, mais aussi et surtout, qu'il faut gérer le cycle de vie de l'analyseur dans un atelier logiciel qui permette une évolution incrémentale et stable sur plusieurs années. Dans cette optique, il est essentiel que le temps de travail ne soit pas gâché dans une régression et que les objectifs de qualité en termes de rappel et précision soient correctement ciblés et atteints.

Abstract

The paper is written in the context of an applied private research approach. After having noticed that the combination of a good lexicon with a good parser is not enough in order to build an operational industrial parser, we looked for efficient solutions to go further. We demonstrate that first, we need to confront the parsing process to real corpora but secondly and above all, that we must manage the life cycle of the parser within a software workbench that permits a stable and incremental evolution over a period of several years. With this respect, it is is essential to avoid losing time in regression and to reach the quality targets in terms of recall and precision.

1 Introduction

Un corpus annoté manuellement est en général petit mais précis et riche, au contraire d'un corpus brut qui n'apporte que peu d'informations rapportées à chaque phrase, mais qui peut être au contraire de taille gigantesque. Nous allons voir comment il est possible de combiner les particularités de chacune de ces ressources avec le but de construire un atelier de développement pour un analyseur industriel qui traite des milliers de documents par jour.
L'article traite de l'analyse du français pour simplifier la présentation mais TagParser existe en version anglaise avec les mêmes mécanismes. Ajoutons qu'une implémentation prototypique a été commencée en espagnol.

2 Principes

Développer un analyseur automatique de langage naturel d'une qualité suffisante pour qu'il rende des services dans un contexte industriel est une tâche relativement lourde. La difficulté ne réside pas tant dans l'élaboration d'un analyseur qui donnerait un bon résultat dans la majeure partie des circonstances : atteindre une F-Mesure de 55% aux tests Easy / Passage est en fait assez facile[1]. La difficulté est plutôt d'améliorer le score de manière incrémentale quand des insuffisances et problèmes sont rencontrés. Ainsi, passer de 55% à 96% est bien plus complexe. Le principal défi consiste à être certain que ce que l'on pense être une amélioration ne va pas remettre en cause le comportement existant.
Nous pensons que la solution nécessite la mise en place d'un atelier dédié à l'évolution de l'analyseur. Le moteur d'analyse en lui-même, même s'il est central, n'est plus qu'un des composants de cet atelier. En d'autres termes, il faut faire en sorte que les actions d'évolution entrent dans un cycle de vie stable pour faire évoluer les données et les programmes sur une échelle de temps de plusieurs années.

3 Corpus

Même si l'on parle couramment une langue, il est extrêmement difficile d'en écrire une grammaire surtout si cette dernière doit tenir compte des exceptions et des formes idiomatiques. Il faut donc procéder par approximation. Pour ce faire, l'outil de départ est le corpus représentatif de la langue en question, même si l'on est certain qu'il ne sera jamais assez vaste.

[1] Voir le chapitre sur la qualité des résultats.

Un gros corpus a été collecté[2] avec les données suivantes :
- une année de dépêches de l'AFP (2007),
- une année du journal Le Monde (1994),
- deux années de dépêches de l'ATS (1994 et 1995),
- wikipédia dans sa version d'avril 2006,
- CR du Parlement Européen (1996 à 2003),
- des règlements communautaires JRC-Acquis,
- deux années de l'Est Républicain (2002 et 2003),
- 1000 recettes de cuisine,
- 1000 articles sportifs et financiers datés de 2008,
- 200 textes littéraires du XIX et XX^{ième} siècle,
- quelques corpus oraux et courriels.

Le corpus comporte 600 millions de mots codés en UTF8. C'est un corpus dit "brut" dans le sens où il ne comporte aucune annotation. Il n'est pas très équilibré selon les genres au sens de Gougenheim [Gougenheim], en effet, la presse est sur-représentée. La principale application actuelle de TagParser étant d'analyser la presse toutes les nuits, ce n'est pas très gênant.

4 Les stratégies possibles

Les ressources pour développer un analyseur sont :
- les règles. Leur écriture est aisée dans un premier temps mais la tâche est plus ardue quand la complexité et l'interdépendance augmentent.
- les connaissances déclaratives lexicalisées [Francopoulo 2006][LMF]. Si leur maintenance est contrôlable, la création est complexe car elle nécessite une abstraction importante si l'on désire atteindre une couverture satisfaisante.
- un corpus annoté qui va permettre d'effectuer un apprentissage. Si les règles de codage sont cohérentes, la maintenance est aisée mais la tâche est laborieuse.
- un corpus brut, qui pris isolément n'est pas d'une grande utilité.
La solution que nous avons choisie consiste à combiner toutes ces ressources en effectuant certaines actions pour améliorer le rappel et d'autres actions pour augmenter la précision.

5 TagParser : la chaîne d'analyse

L'analyseur est de type montant. Il enchaîne les principaux modules suivants sous forme d'un pipeline avec un segmenteur, un analyseur morphologique, un chunker (qui effectue le tagging) et un module de calcul des relations syntaxiques. La sortie d'analyse comporte trois types de résultat: les constituants sans enchâssement (i.e. un chunk), les relations syntaxiques (e.g. la relation sujet) et les entités nommées. La communication d'un module à l'autre respecte les principes du Linguistic Annotation Framework [LAF] [Ide], dans le sens où chaque module ajoute une annotation de type déportée ('stand-off' en anglais) sur la donnée transmise. Les entités nommées selon leur portée (est-ce un mot, un chunk ou un multi-chunks ?) sont calculées par les différents modules. Par exemple: une URL est reconnue par l'analyse morphologique mais une séquence "prénom + nom" est reconnue par le chunker. Les entités nommées plus complexes de type "Bill et Hillary Clinton" avec factorisation du nom de famille sont déterminées après le calcul des relations puisqu'elles nécessitent le calcul de la coordination.
Les règles de formation des structures syntaxiques sont conformes au guide d'annotation PEAS établi initialement dans le projet PEAS avec les évolutions des projets Technolangue/Easy[3] et ANR-Passage[4]. Le format de sortie respecte les standards professionnels spécifiés par l'ISO. C'est le format Passage [Paroubek] [Clergerie] défini à partir de MAF et SynAF [MAF] [SynAF]. Le détail est présenté dans l'illustration-1.

6 L'atelier de développement

Le développement est fondé sur les quatre processus suivants :
- la gestion du cycle de vie du chunker,
- le développement des relations,
- l'apprentissage des affinités de rattachement,
- la vérification post-mortem.

7 La gestion du cycle de vie du chunker: apprentissage actif

L'annotation des données est un pré-requis pour un grand nombre de programmes de TAL. Malheureusement, produire des données fiables en grandes quantités est une tâche laborieuse. Il a été montré que l'apprentissage actif peut être employé pour en réduire le coût sans baisse de qualité[5]. Concrètement, il s'agit de combiner un corpus annoté avec un corpus brut par sélection dynamique de fragments. L'annotation est effectuée à la main, phrase par phrase. A partir du contenu textuel, il s'agit d'ajouter les informations de frontière de chunk, l'étiquette du chunk et les étiquettes morphosyntaxiques de chaque mot. A l'état initial, avec un corpus vide, supposons que nous ajoutions la phrase: "Albert marche.", l'annotation sera la suivante sur deux lignes :

```
[Albert GN]  [marche GV]    .
prénom       verbeConjugué ponctuationFinale
```

[2] Nous remercions vivement François-Régis Chaumartin (Proxem), Denis Teyssou (AFP), Eric de la Clergerie (INRIA-Alpage), Frédéric Marcoul (Spotter) et Bertrand Gaiffe (ATILF) pour leur aide sur les corpus.

[3] http://www.technolangue.net/article198.html

[4] http://atoll.inria.fr/passage

[5] Voir par exemple, l'atelier Active Learning de NAACL-2009 sur http://nlp.cs.byu.edu/alnlp

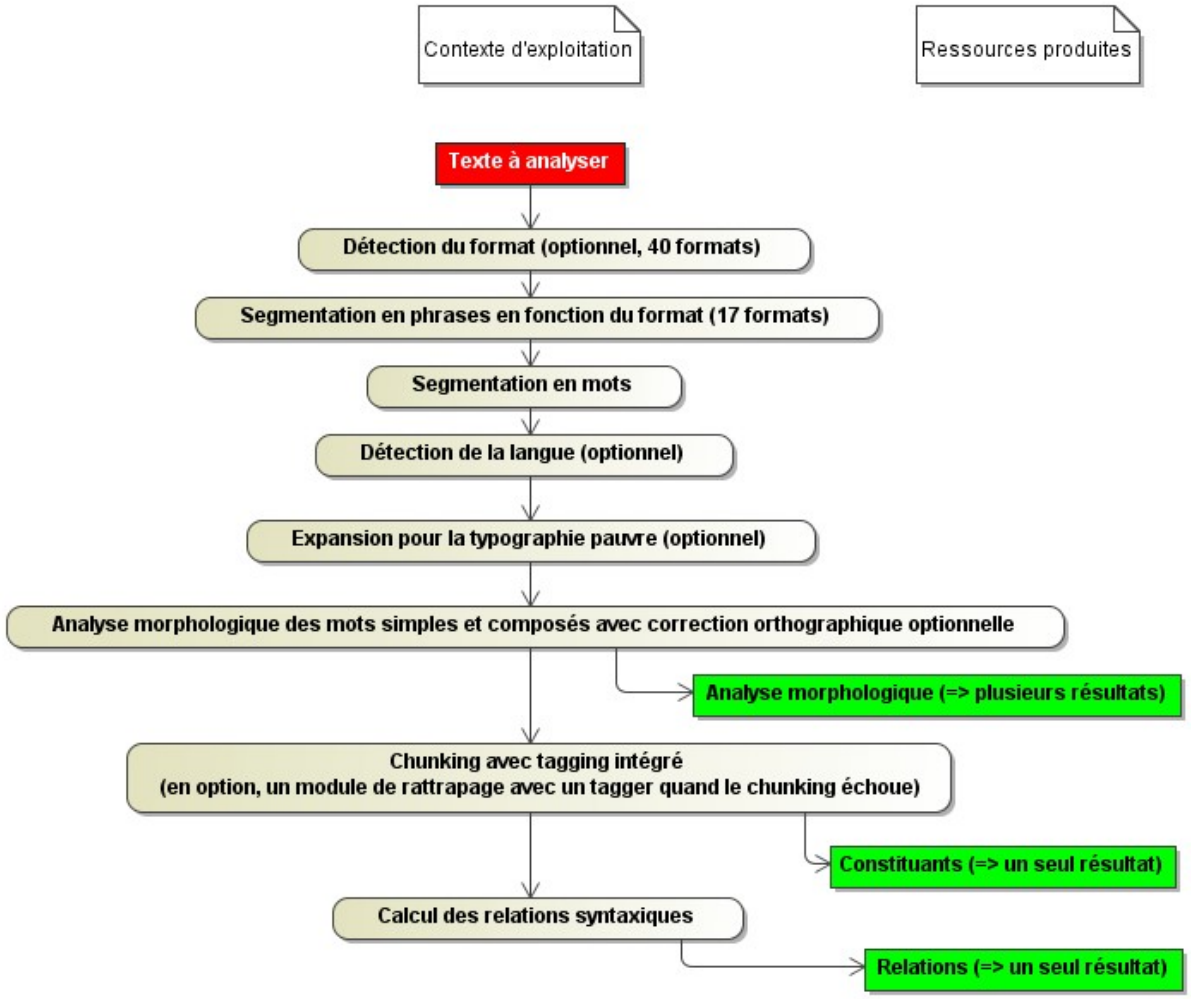


Illustration-1: Chaîne d'analyse

Le corpus ne possède donc qu'une seule phrase. Un programme d'apprentissage par induction [Francopoulo 2003] peut alors construire automatiquement un chunker qui est capable de reconnaître "Albert marche" mais aussi "Adrienne dort". Supposons maintenant que nous confrontions ce chunker à la phrase "Adrienne dort profondément.". Le chunker (qui est lancé sans mode de rattrapage) échoue dans son analyse puisqu'il n'a, à cette étape du développement, aucune connaissance des adverbes. L'échec d'analyse est enregistré dans un fichier. Il faut alors annoter manuellement cette phrase qui va faire grossir le corpus annoté. D'un cycle à l'autre, le dispositif permet de construire un chunker du français.

Dans le détail, les mécanismes sont un peu plus complexes en trois points:

- l'induction comporte une phase de vérification de la cohérence des annotations. Il est vérifié que le chunker induit donne effectivement les mêmes annotations que celles qui figurent dans le corpus annoté.
- les échecs sont triés par ordre de complexité croissante, ce qui permet de cibler la configuration de l'échec et évite ainsi d'annoter des fragments non pertinents. Ce point est important car le caractère le plus délicat de l'apprentissage actif est la sélection des exemples à chaque itération. Il s'agit de choisir les phrases dont le retour sur investissement (effort rapporté à l'apport au système) est le plus élevé. Une difficulté supplémentaire est que, même si la couverture est la faiblesse la plus fréquente, d'autres phénomènes parasitent le dispositif comme les fautes d'orthographes, l'absence lexicale ou divers problèmes de formatage textuel. Il faut donc une opération manuelle pour distinguer ces situations. Nous posons deux hypothèses qui sont, pour la première, que l'effort est proportionnel au nombre de mots et pour la seconde, que plus une phrase est petite plus elle a de chances d'exhiber un phénomène nouveau rapporté à l'effort. En conséquence, à chaque itération, nous annotons la plus petite phase en échec.
- sans qu'il soit nécessaire de développer ici, le chunker possède un mode de rattrapage par tagging via des trigrammes. Ce mode n'est utilisé qu'en phase d'exploitation, pas en développement. Ces trigrammes sont calculés en même temps que l'induction.

Le cycle de vie du développement est schématisé par le diagramme UML d'activité suivant:

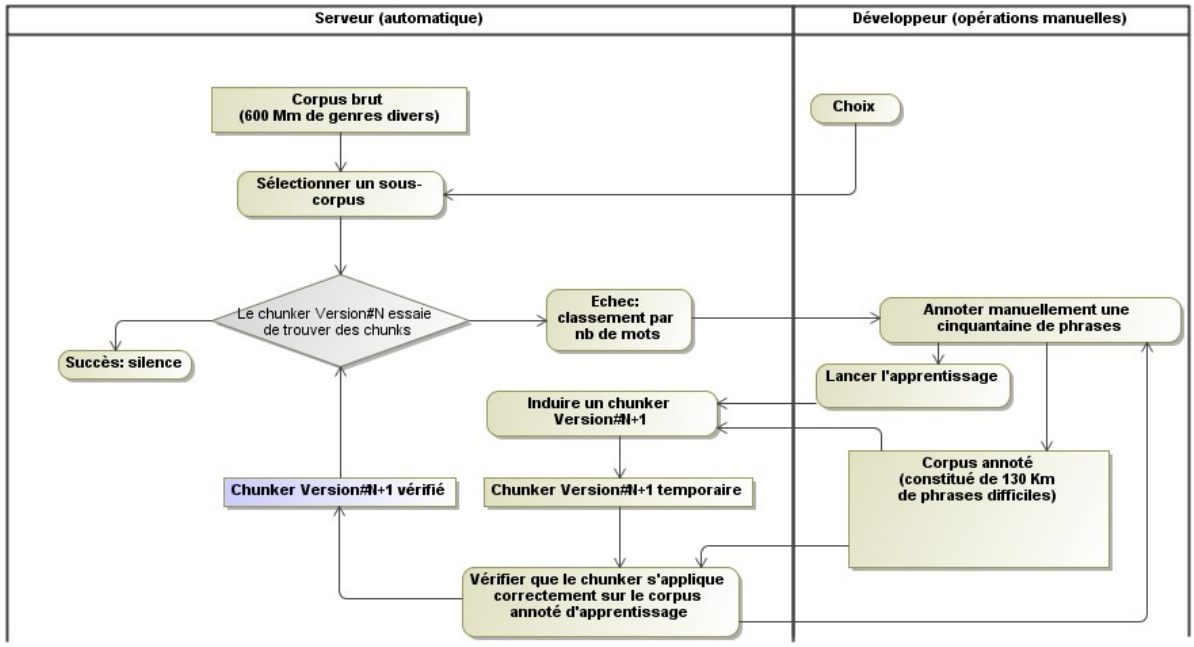


Illustration-2: Cycle de vie du chunker

Sur le français, le corpus annoté comporte actuellement 130 K mots et le corpus brut 600 M mots. L'annotation (manuelle) est longue et pénible, même si elle est incrémentale. Dans la pratique, au lieu de procéder sur la totalité du corpus brut, la personne chargée du développement sélectionne un sous-corpus. Il est ainsi possible d'orienter le développement en fonction de tel ou tel texte ou genre. Actuellement, le chunker n'est pas capable d'analyser correctement tous les phénomènes présents dans le corpus brut, même s'il n'en est pas loin. Selon les textes présentés, le taux d'échecs (sans mode de rattrapage) oscille entre 2% et 4%. Notons que ce taux n'est pas simplement dû à des insuffisances du chunker mais aussi à quelques phénomènes marginaux qui s'ajoutent comme des fautes d'orthographe mal redressées ou des problèmes de mise en page.

Le corpus annoté est en réalité un corpus de difficultés puisqu'il est le résultat de l'accumulation d'échecs d'analyse[6]. Ce n'est pas un corpus dont la distribution statistique serait celle d'un corpus pris au hasard. La distribution est biaisée. Mais en contre-partie, la tâche d'annotation est bien plus légère que si les phrases avaient été choisies au hasard.

Pour finir, on notera que le dispositif permet d'un cycle à l'autre d'améliorer le rappel puisque la couverture est augmentée des phénomènes qui provoquaient un silence lors du cycle précédent. En revanche, le dispositif ne prouve pas que toutes les analyses sont bonnes. La vérification prouve simplement que les analyses sont cohérentes entre elles. La précision sera contrôlée partiellement par l'outil de vérification port-mortem (voir plus loin).

8 Le développement des relations

Une fois les constituants calculés, il s'agit de déterminer quelles sont les relations syntaxiques.

La gestion n'est pas effectuée par apprentissage mais par édition manuelle de règles avec contrôle par test de non-régression.

Le jeu d'étiquettes des relations comporte les 14 valeurs du guide d'annotation Passage, avec par exemple "sujet" ou "modifieur de nom". Le calcul combine un lexique et des règles. Pour le lexique, des informations de rection issues de DicoValence [Van den Eynde] sont associées à des indications de contrôle issues de Lefff [Sagot] et des propriétés syntaxiques décrites par Tagmatica.

Les règles sont écrites à la main. A chaque fois qu'une modification est effectuée, un programme de comparaison est lancé afin de vérifier que le comportement n'est pas dégradé. Le corpus de référence utilisé est constitué des exemples du guide d'annotation augmenté de l'historique des tests. Le programme de comparaison comporte de nombreuses options sophistiquées pour filtrer ou orienter les affichages. Ce dispositif permet de contrôler à la fois la variation en rappel et en précision.

La gestion des relations est représentée par l'illustration-3, de même que les mécanismes du chapitre suivant :

[6] Notons à ce propos que 130K mots de difficultés est une taille très importante. C'est évidemment bien plus vaste au plan de la diversité des phénomènes que 130K mots pris au hasard.

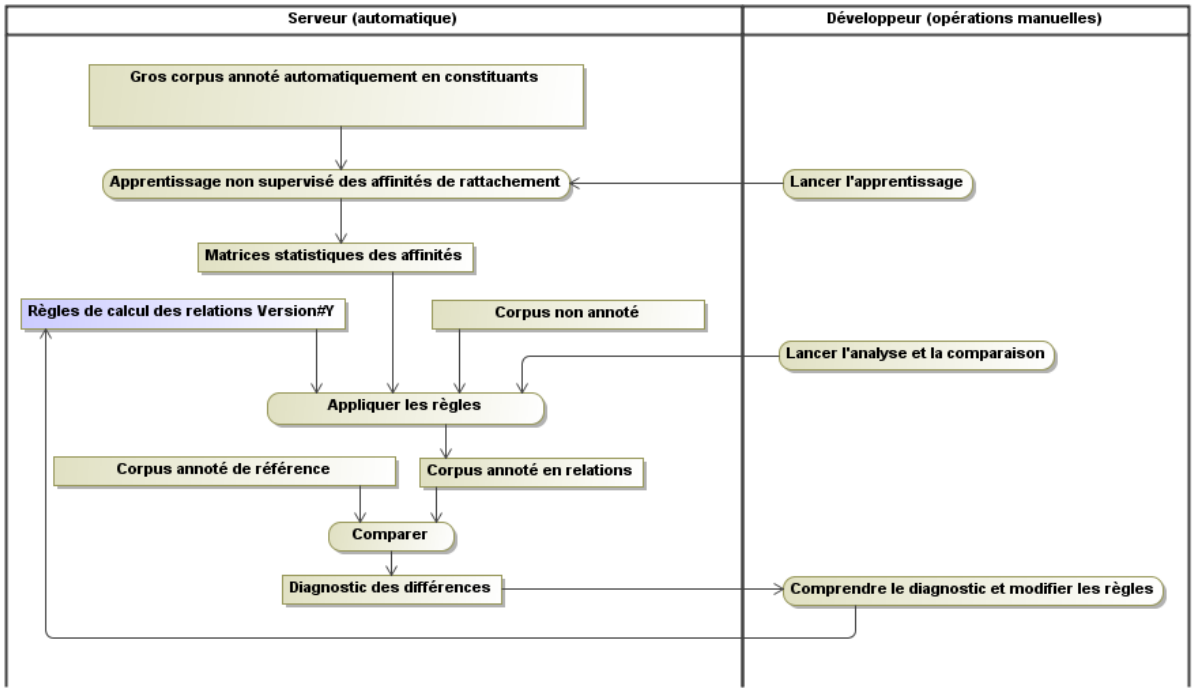


Illustration-3: Gestion des relations et affinités

9 L'apprentissage d'affinités de rattachement

Une fois que le chunker est suffisamment développé et stable, il est possible de s'intéresser aux rattachements entre groupes.
Par exemple, il s'agit de déterminer quels sont les triplets nom#1 / préposition / nom#2 qui figurent dans un corpus dans une situation où l'on peut déterminer de manière certaine que le nom#2 est un modifieur du nom#1 [Bourigault]. Une telle situation est le début de phrase GN+GP qui signifie de manière quasi-certaine en français que le GP modifie le GN. Ce sera par exemple: "Le chien de sa mère ..." dans lequel "de sa mère" est un modifieur de "le chien".
Un programme d'apprentissage non supervisé opère à partir de l'analyse en constituants du gros corpus. Le résultat est une matrice de poids qui seront exploités pour favoriser tel ou tel rattachement lors du calcul des relations. Le manque de place ne permet pas de donner beaucoup plus de détails sur ce mécanisme, mais disons simplement que l'apprentissage calcule aussi des affinités de rattachement sur le couple sujet / verbe. On observe que le dispositif est orienté exclusivement vers l'amélioration de la précision : il permet de gagner 7% selon les critères énoncés au chapitre "Qualité des résultats". Le rappel n'est pas concerné.

10 La vérification post-mortem

Il s'agit de lancer la chaîne d'analyse au complet et de s'assurer que la sortie respecte bien certaines propriétés formelles définies dans le guide d'annotation. Par exemple, il s'agit de vérifier qu'un GP commence effectivement par une préposition ou qu'un noyau verbal comporte un verbe et un seul. Le système comporte une quarantaine de règles comme celles-ci. Elles sont évaluées périodiquement sur les portions du gros corpus sur lesquelles l'apprentissage actif a été appliqué. Le dispositif permet de gérer la précision.

11 Postraitements sémantiques

Sans développer en détail ici, l'analyse syntaxique est complétée par trois modules : le regroupement de variantes (à la Fastr), la résolution des références pronominales et l'extraction des citations.
Il n'y a aucun dispositif automatique mais les résultats sont exploités par des utilisateurs et des développeurs qui de temps à autre effectuent des remontées d'anomalies. Une opération de correction sur l'analyse syntaxique est alors entreprise.

12 Mise en oeuvre

La totalité du code a été écrit en Java de manière portable et industrielle sur une période de sept ans.

Les différents modules sont des packages Java qui s'enchaînent via des appels en mémoire afin d'ajouter une annotation selon la stratégie LAF comme indiqué au chapitre-5. Mais l'implémentation est optimisée: il est notamment hors de question qu'un module écrive son résultat sur disque et qu'un autre module prenne la suite en relisant sur disque. C'est beaucoup trop lent.

De plus, une grande attention a été portée à la factorisation du code via une conception objet avec

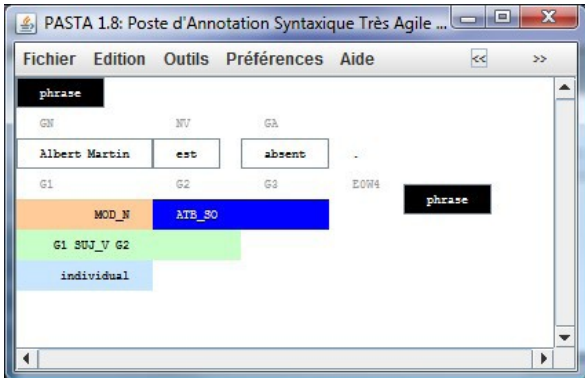
héritage afin de minimiser le code entre les langues, les formats de fichiers ou de manière générale tout code qui peut être partagé entre les modules.

TagParser fonctionne sur Linux et Windows en mode 32 bits ou 64 bits. Le débit d'analyse est d'environ 1,3M mots par heure, ce qui est ni très rapide, ni très lent. Ainsi, pour analyser les 100M mots de la campagne Passage#2, il faut 3 jours de calcul sur un PC Nehalem.

Sur la phrase « Albert Martin est absent », la sortie XMLsera:

```
<?xml version="1.0" encoding="UTF-8"?>
<Document dtdVersion="2.0" file="test.txt">
		<MSTAG id="nS"><fs><f name="grammaticalNumber"><symbol
		value="singular"/></f></fs></MSTAG>
		<MSTAG id="nP"><fs><f name="grammaticalNumber"><symbol
		value="plural"/></f></fs></MSTAG>
		<MSTAG id="nX"><fs><f name="grammaticalNumber"><vAlt>
				<symbol value="singular"/><symbol
				value="plural"/></vAlt></f></fs></MSTAG>
		<MSTAG id="gM"><fs><f name="grammaticalGender"><symbol
				value="masculine"/></f></fs></MSTAG>
		<MSTAG id="gF"><fs><f name="grammaticalGender"><symbol
				value="feminine"/></f></fs></MSTAG>
		<MSTAG id="gE"><fs><f name="grammaticalGender"><vAlt><symbol
				value="masculine"/><symbolvalue="feminine"/></vAlt>
				</f></fs></MSTAG>
		<MSTAG id="mI"><fs><f name="verbFormMood">
				<symbol value="infinitive"/></f></fs></MSTAG>
		<MSTAG id="mP"><fs><f name="verbFormMood">
				<symbol value="participle"/></f></fs></MSTAG>
		<MSTAG id="mC"><fs><f name="verbFormMood">
				<symbol value="conjugated"/></f></fs></MSTAG>
		<MSTAG id="mU"><fs><f name="verbFormMood">
				<symbol value="unknown"/></f></fs></MSTAG>
<Sentence id="E0" trust="100">
		<T id="E0W0T0" start="0" end="6">Albert</T>
		<T id="E0W1T0" start="7" end="13">Martin</T>
<G id="E0G1" type="GN">
		<W id="E0W0" tokens="E0W0T0" pos="properNoun"
				lemma="Albert" form="Albert" mstag="gM"/>
		<W id="E0W1" tokens="E0W1T0" pos="properNoun"
				lemma="Martin" form="Martin"/>
</G>
		<T id="E0W2T0" start="14" end="17">est</T>
<G id="E0G2" type="NV">
		<W id="E0W2" tokens="E0W2T0" pos="verb"
				lemma="être" form="est" mstag="mC"/>
</G>
		<T id="E0W3T0" start="18" end="24">absent</T>
<G id="E0G3" type="GA">
		<W id="E0W3" tokens="E0W3T0" pos="qualifierAdjective"
				lemma="absent" form="absent"/>
</G>
		<T id="E0W4T0" start="24" end="25">.</T>
		<W id="E0W4" tokens="E0W4T0" pos="mainPunctuation"
				lemma="." form="."/>
<R id="E0R0" type="SUJ-V">
		<sujet ref="E0G1"/>
		<verbe ref="E0G2"/>
</R>
<R id="E0R1" type="ATB-SO">
		<attribut ref="E0G3"/>
		<verbe ref="E0G2"/>
		<s-o valeur="sujet"/>
</R>
<R id="E0R2" type="MOD-N">
		<modifieur ref="E0W0"/>
		<nom ref="E0W1"/>
</R>
<NE id="E0E0" type="individual" lst="E0W0 E0W1"/>
</Sentence>
</Document>
```

Comme ce n'est pas très lisible, un outil d'affichage est utilisé:



La partie haute présente les constituants. Ensuite, nous avons les relations modifieur de nom entre le prénom et le nom de famille, l'attribut du sujet et enfin la relation sujet entre le groupe G1 et G2. La dernière ligne indique le marquage de l'entité nommée « Albert Martin » qui est de type « individual » (i.e. nom de personne).

13 Qualité des résultats

Sur le français, la plate-forme de référence pour l'évaluation d'un analyseur est le site du projet ANR-Passage hébergé par Elda [Hamon 2008]. Il faut demander l'accès au corpus d'évaluation puis soumettre le résultat d'analyse et attendre 10 mn que le service web donne les notes. Quand on lance TagParser avec le mode de rattrapage, on obtient 96% en F-Mesure pour l'analyse en constituants et 66% pour les relations, ceci lors de l'été 2009.

On observe que TagParser est meilleur pour les constituants comparativement aux relations. Ce résultat est compréhensible dans la mesure où une plus grande énergie a été consacrée à cet aspect suite à des demandes précises de clients.

Notons de plus que la reconnaissance des entités nommées donne satisfaction. Une étude d'évaluation a été menée mais il est délicat d'en donner les détails car d'une part elle ne portait que sur un type de presse et d'autre part, elle n'était pas quantifiée.

On peut se poser la question d'évaluer quelles sont les limites hautes des différents systèmes d'apprentissage. Ce n'est pas facile à prévoir de manière certaine. Actuellement, nous observons que les phrases sélectionnées par l'apprentissage actif sont fréquemment des tournures rares ou idiomatiques, qui donc dans ce dernier cas, passent par l'ajout d'un mot composé plutôt que par l'augmentation de la couverture syntaxique. Il semble donc que la limite soit presque atteinte. Il n'en est pas de même pour les relations puisqu'il nous arrive de travailler quelques heures et de faire varier la F-Mesure de quelques

centièmes, ce qui est beaucoup. Notons à ce propos que dans la mesure où nous opérons sur de gros corpus relativement divers, nous sommes à l'abri des phénomènes de sur-apprentissage.

14 Conclusion

Dans cet article, sur un exemple pratique, nous avons essayé de montrer qu'il était à la fois important mais aussi possible de concevoir des outils qui permettent des évolutions incrémentales. Il faut se confronter aux corpus réels et intégrer le cycle de vie de toutes les ressources pour obtenir un système stable.

Références

Communications académiques

Bourigault D., Frérot C 2004 Ambiguïté de rattachement prépositionnel: introduction de ressources exogènes de sous-catégorisation dans un analyseur syntaxique de corpus endogène. TALN Fès.

Clergerie (de la) E., Ayache C., Chalendar (de) G., Francopoulo G., Gardent C., Paroubek P. 2008 Large Scale Production of Syntactic Annotations for French. Workshop on syntactic annotations for interoperable language resources. ICGL Hong Kong.

Francopoulo G. 2003 TagChunker: mécanisme de construction et évaluation, TALN Batz.

Francopoulo G., George M., Calzolari N., Monachini M., Bel N., Pet M., Soria C. 2006 Lexical Markup Framework (LMF), LREC Genoa.

Gougenheim G. 1958 Dictionnaire fondamental de la langue française, éditions Didier.

Hamon O., Paroubek P., Mostefa D. 2008 SEWS: un serveur d'évaluation orienté Web pour la syntaxe, TAL vol 49-2.

Ide N., Romary L. 2006 Representing linguistic corpora and their annotations. LREC Genoa.

Paroubek P., Clergerie (de la) E., Loiseau S., Vilnat A., Francopoulo G. 2009 PASSAGE Syntactic Representation. TLT Groningen.

Sagot B. LEFFF, lexique des formes fléchies du français. http://alpage.inria.fr/~sagot/lefff.html

Van den Eynde K., Mertens P . Dicovalence, dictionnaire de valence des verbes français, http://bach.arts.kuleuven.be/dicovalence

Standards publiés ou en développement

LAF (en développement) Représentation des annotations: ISO-24612 Linguistic Annotation Framework, ISO, Genève.

LMF (publié) Standard ISO de représentation des lexiques: ISO-24613 Lexical Markup Framework, ISO, Genève.

MAF (en développement) Représentation de la morphosyntaxe: ISO-24611 Morphosyntactic Annotation Framework, ISO, Genève.

SynAF (en développement) Représentation de la syntaxe: ISO-24615 Syntactic Annotation Framework, ISO, Genève.